Zapr Audio Fingerprinting

Harikrishnan Potty
AVP - DSP
Zapr Media Labs
Bangalore, India
Email: harikrishnan@zapr.in

Srikanth Konjeti
AVP - DSP
Zapr Media Labs
Bangalore, India
Email: srikanth@zapr.in

***Abstract*—The problem of audio fingerprinting translates to finding the right piece of audio, given a noisy version of the same. In this short paper we provide the outline of a noise robust audio fingerprinting and matching algorithm based on the local spectrogram energy peaks.**

I. Introduction

The task of audio fingerprinting is to find the correct piece of audio (also refereed to as the reference audio), given a noisy version of the same (also refereed to as query audio). The fingerprinting algorithm should be robust to noisy environments, differences in the number of audio channels, differences in audio encoding and should also be able to cater to a wide variety of mobile phone microphones (assuming that the query comes from a mobile phone, which is generally the case).Also the query audio can be from any part of the complete reference audio file.

In addition to it, the fingerprint representation should be as unique and as compact as possible, allowing us to declare match to a query file in real time.Some of the initial work of audio fingerprinting can be found in [1],[2].

The Audio Fingerprinting and Matching algorithm outlined here consists of two basic components. The first one is the database builder which takes in a set of Reference (Clean) audio files, fingerprints them and builds a database.

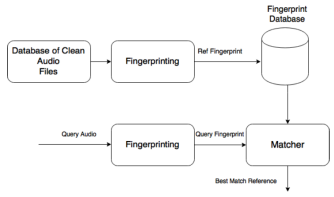

Fig. 1: Generalized Audio Fingerprinting

The Second part is the query matching stage which takes a noisy query file, computes the fingerprint, matches the query against the database and returns the correct match.

II. Algorithm Outline

The algorithm that we employ is based on finding local energy maximas in a spectrogram and performing combinatorial hashing as described by Wang[1]. The input audio signal is downsampled to 8 KHz and is divided into frames of length 512 samples with 50% overlap between the consecutive frames. From each of the frames, we select energy peaks which we believe are robust to signal degradation and combine these points across multiple frames. These combination of points across multiple signal frames are refereed to as audio hashes.The Final fingerprint output that we obtain are these hashes and the time in which they occur.

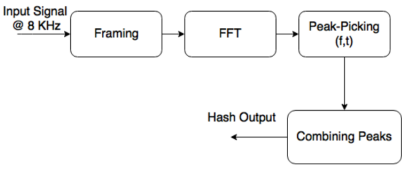

Fig. 2: Audio Fingerprinting algorithm

III. Fingerprint Matching

Once the query fingerprint is computed, a time aligned matching is performed where the sequence of query hashes in the fingerprint is matched against all the reference hashes. The reference which matches the query with the hits more than a threshold is declared as the match for that query.The Offset is the delay of the query with the reference.

IV. Implementation

Both the Fingerprinting the matching algorithms are implemented in CPP. The binaries are provided in the submission directory. Two python scripts in accordance with the submission format are also provided to run the binaries.

V. Folder Contents

The Submission Folder contains of the following programs/documents:

- **Builder.py:** Python script for constructing the database.
- **Matcher.py:** Python wrapper for invoking the matcher binary on the query fingerprint files.
- **Fingerprinting.py:** Python wrapper for invoking the fingerprinting binary on audio files.
- **Indexer:** CPP Binary for generating the audio hashes.
- **Matcher:** CPP Binary which performs the actual matching.
- **README.txt:** A readme file which contains the details on how to run the scripts, details regarding the necessary command line arguments and details regarding the build system for compiling the binaries.

REFERENCES

[1] A.Wang, *The Shazam music recognition service*, 48, Aug. 2006.

[2] J. Haitsma, T. Kalker, and J. Oostven *Robust audio hashing for content identification*, in Procs. of the International Workshop on Content-Based Multimedia Indexing, (Brescia, Italy), October 2001.